4/11/2024

Rule of Thirds AI

Authors:

Mohammad Al-hemed: 101154860
Ali Baobaid: 101160656

Abstract:

This project took a lot of hard, arduous work, and a lot of time. It took many steps, all of which combined together allowed us to compile our final rule of thirds, image processing AI. We started off by creating a blueprint for what we needed to do and we determined how we wanted to do what we needed to do. With that being said, we took into consideration some of the techniques used in the past couple of assignments including opencv, and matplotlib. Our project primarily made use of opencv techniques including loading the image (imread), drawing grid lines (cv2.line), saving the image (imwrite), and cvtcolor which is used in contingency with matplot since cv2 and matplot use different formatting (RGB and BGR). We also primarily used matplotlib throughout the code, as it proved immensely useful. For instance, pyplot.subplots is used to create a plot and subplots (original and modified), set_title is used for adding titles to the plots, and .show allows us to display the results. As can be seen, we implemented a lot of what was learned in class, and through the past assignments, as they were integral in our succession. They proved to be the most efficient, and after trial and error, they made the most logical sense to use.

The way we structured and implemented the code took many steps. That being - to first create a grid which would overlay the image which is what the first function does. This grid is a 3x3 matrix which splits the image up into 9 different images. With three columns, and three rows, the rule of thirds overlay was complete, however it was necessary to calculate the position of these lines. This is vital since each image has different dimensions and ratios, and so we had to research methods of which we could determine these. This was done by using cv2.line, and allowed us to implement the overlay overtop the image. Once this was done, the next function we implemented was the actual implementation of the rule of thirds over the image. The image is loaded in with .imread, calculates the coordinates needed to determine the center of the image, and aligns it with the previously calculated rule of thirds overlay using cv2.line, and crops the image based on that. It is important to note that the image path is required, and depends on the individual, and it is necessary to implement this as required. Currently the image is in the same folder, and all that is required is to paste that path into image_path, for the function to work properly. Finally, the code uses matplotlib to plot the two images, one being the result and one being the new modified image, along with a title. This is a general overview of the functions of the code, and explains the fundamentals of our blueprint.

Introduction:

The project we decided to work on is an artificial intelligence auto analysis that implements the rule of thirds which is a technique that is heavily relied upon in photography. The reason we decided on this is because while it is a technique used in taking actual pictures, we thought it would be interesting to implement this as an image processing technique. Rule of thirds in photography is primarily used as a tool to raise the quality and eye candy of an image, and so an AI which does this automatically would greatly improve the photo, with such a simple method, instead of going back and retaking the photo. The goal is to analyze the photo, apply the rule of thirds based on the subject, adhering to the 9 column split, and ensuring the column/row falls within the intersection/lines.

Key challenges include deciding how to calculate points of interest for the rule of thirds. This could be done with a plethora of methods including calculating contrast, brightness, edge detection, facial recognition, etc. Figuring out the most convenient, logical, and efficient method to determine how to find the subject, and implement cropping or zooming in was difficult. This brings up the next issue which was if the subject was all the way to the left, right, top, bottom. When taking a photo, you just need to shift the camera; however with image processing, it is required to copy the column/row and paste it to fit the rule of thirds. Other options included zooming in. It was also a challenge to locate the subject, but the method that was researched was face recognition, however, this only works for people. We decided on lambda (mentioned in the textbook) which finds the closest intersection points in the image by using min() and cropping the photo based on that depending on image dimensions and aspect ratios.

Background:

When completing this project, we completed a lot of the research within our own scope of what we could find, and that included analyzing some prior works that were used in the past. For starters, we needed a basic understanding of how the rule of thirds works. “When you take the point of interest and move it to one of these intersection lines, it creates more of a dynamic image. It allows the eyes to come into the frame, look at the object, look at the background, have a more natural flow, and connect to what they see.” (Alex 2023).

We also looked at some potential ideas for how certain implementations were made, but nothing outside the scope of what we learned in class. For instance, a lot of the implementations learned in class were incorporated like from assignments 1 and 2, including cv2, and matplotlib, as well as some test runs with numpy. One source for cropping images was useful, and simple as it required .imread, .shape, as well as slicing the image, which was incorporated in class as well. (Ayushama 2023)

One specific paper that allowed us to progress in the development of the rule of thirds AI was Rule of Thirds Detection from Photograph by Long Mai. It deals heavily with saliency, facial, and object recognition, however as mentioned below, this is a highly efficient method that takes a lot more time. Certain techniques and methodologies used in the progression in regards to saliency require certain functions, certain knowledge, as well as certain cases which must be hard coded. To maintain complexity and time efficient programming, we decided to lean towards image ratios and dimensions, despite the useful compatibility it is able to accomplish. "We design a variety of features based on the saliency and objectness map to detect visual elements and infer the spatial relationship among them. (Mai n.d).

Finally, we used a lot of what was learned in class, but what pertained most to our project was lectures 8 and 9. These had to do with locality, saliency, image alignment, and transformations. These are integral in our implementation as they provide the basis for our rule of thirds overlay, and understanding these provides a well structured basis for our goal. (Komeili 2024).

Methodology:

As previously mentioned, we had a couple of different ideas about how we would go forward with this however after many tries, we decided to go with what was simplest and made the most sense. We tried edge detection, brightness detection, contrast detection, facial recognition, object detection, and even user interaction. The key similarity between all of these is that they were time intensive, difficult, and required a lot more research. Brightness detection and contrast detection took a lot of effort to implement and provided us with inaccurate results. This had to do with the fact that a lot of actual implementation of the rule of thirds had to be hard coded. By that - I mean that each image is different, and so determining the contrast and brightness may work for some images, and may be the complete opposite for others. This includes very bright images, very dark images, high contrast images, etc. This can definitely be implemented however requires more tedious code. The same can be said for facial recognition, object recognition, and user interaction which deal with implementations and libraries including detectmultiscale, and readnet(). This required a lot more research when incorporating them into the rule of thirds code, compared to brightness and contrast, which we had running, but resulted in many failures.

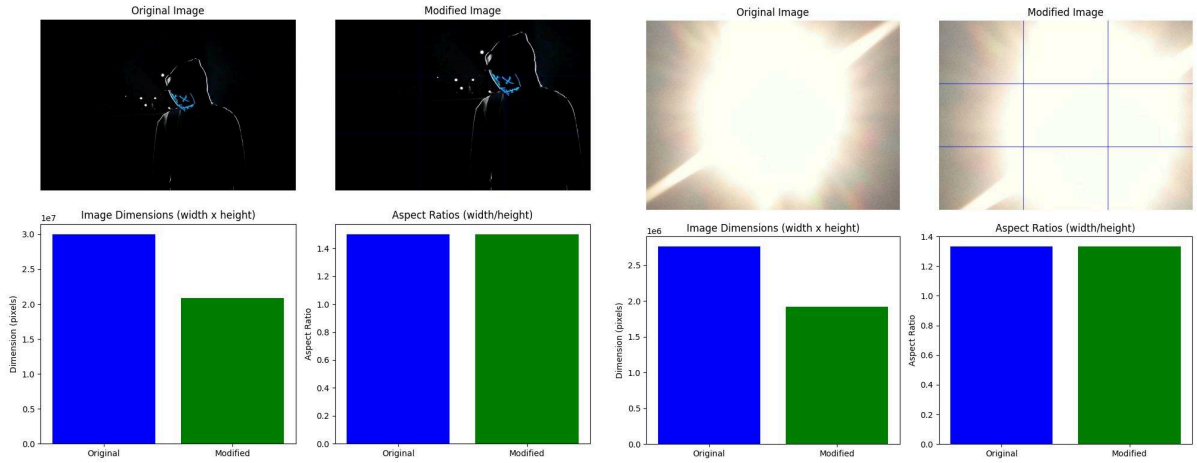
With that, we figured to go off image analysis and image ratios/dimensions. The reason for this is that it provided us with more accurate, and consistent results, after much trial and error of the above. It implies the project mission into a simple, neatly formatted method that works for every image, and creates a basis that remains constant with each image, despite the brightness, contrast, gradient, scenery, color, and subject (object vs person).
As can be seen below, each image varies in regard to dimensions and ratios which is why we need cv2.line. An incorrect implementation would fault the image because each image is

different and they come in many different sizes. The modified images maintain a constant dimension and ratio.

Data set(Check file in folder):
Image dimensions for 10 photos: boat 1: 50000, tester3: 4500000, love:500000 female: 50000, chipmunk: 50000, tester1.jpg: 50000, suited: 50000, marriage: 1.7 blueman: 3.0 bright: 2.8
Image ratios for 10 photos:1.75 1.5 0.8 1.75 1.5, 1.5 1.5 0.65 1.5 1.3

Image dimensions for 10 modified photos:35000, 300000, 350000, 350000, 350000, 350000, 35000, 1.1, 2.0, 1.8
Image ratios for 10 modified photos: 1.75, 1.5 0.8 1.75 1.5, 1.5 1.5 0.65 1.5 1.3

Results:

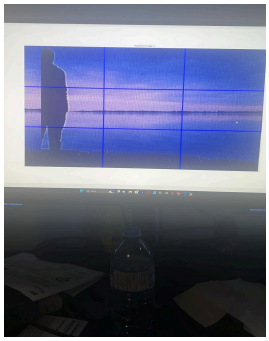


Key findings throughout the project are that the image dimensions and ratios are affected with every inputted image. Dimensions tend to get smaller while ratios tend to get higher. This makes sense and shows that our code is working properly. Since the rule of thirds crops the image, the hypothesis is that the dimensions would drop while the ratio would rise. This is shown above, as well as in the data sets. With that being said, the image is cropped, and the rule of thirds is properly implemented with every image. Certain exceptions are listed below in the conclusion.

Conclusion:

To conclude, our rule of thirds AI does what we want. We modify the photo so that it is cropped and the rule of thirds is applied. This is done through image analysis of the image dimensions and ratio. This strays away from saliency at the cost of providing a more efficient approach. Unfortunately, we were met with certain exceptions which are if the photo is already in the rule of thirds then the image tends to stay the same. Another would be if the subject is all the way to the right/left/top/middle. When this happens, a form of rule of thirds is applied, which is the

zooming in of a photo, allowing the subject to be within the grid lines, however, it would match the rule of thirds more closely if it was shifted to the right. Unfortunately, this would require more research, however, we took one approach which ultimately failed in the photo below. To copy the row/column to the right and paste it to the adjacent side of the subject. This resulted in many failures but is one option to explore in the future.



References

Alex. (2023). Rule of Thirds: The Composition Rules. Enterprise DNA Blog. Retrieved from https://blog.enterprisedna.co/rule-of-thirds-the-composition-rules/

Ayushama. (2023). Crop Image with OpenCV-Python. GeeksforGeeks. Retrieved fro https://www.geeksforgeeks.org/crop-image-with-opencv-python/

Komeili, M. (2024). Computer Vision: Image Features. Brightspace by D2L. https://brightspace.carleton.ca/d2l/le/content/220988/viewContent/3654562/View

Komeili, M. (2024). Computer Vision: Homography and Image Alignment. Brightspace by D2L. Retrieved from https://brightspace.carleton.ca/d2l/le/content/220988/viewContent/3659439/View

Mai, L. (n.d.). A Concise Approach to Mathematical Analysis of Gabor-like Representations and Algorithms. Portland State University. Retrieved from https://web.cecs.pdx.edu/~fliu/papers/ism2011.pdf

Persson, M. (2018). Lambda Twist. ECCV 2018. Retrieved from https://openaccess.thecvf.com/content_ECCV_2018/papers/Mikael_Persson_Lambda_Twist_An_ECCV_2018_paper.pdf

Szeliski, R. (2021). Computer Vision: Algorithms and Applications (2nd ed.). Retrieved from https://szeliski.org/Book/download/FEGWA4NOZFAE/Szeliski_CVAABook_2ndEd.pdf